UDC: 004.89 https://doi.org/10.15407/jai2023.02.056

ВИКОРИСТАННЯ НЕЙРОЕВОЛЮЦІЇ ПРИ ПОШУКУ ПОЛІТИК В ФОРМІ НЕЙРОМЕРЕЖ ДЛЯ УПРАВЛІННЯ РОБОЧОЮ КІНЦІВКОЮ

А. Є. Вітюк[1], А. Ю. Дорошенко[2]
[1,2]Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Україна
Пр. Перемоги, 37, м. Київ, 03056
[2]Інститут програмних систем НАН України, Україна
Пр. Академіка Глушкова, 40, м. Київ, 03187
[1]alyonavityuk@gmail.com
[2]doroshenkoanatoliy2@gmail.com
[1]https://orcid.org/0000-0002-1445-9598
[2]https://orcid.org/0000-0002-8435-1451

Анотація. Розглянуто підхід до використання нейроеволюції для пошуку політик нейронної мережі для завдання позиціонування роботизованої руки. Робототехнічні задачі, як правило, мають відносно великі простори розв'язків, тому тут нейроеволюційні алгоритми виступають гарною альтернативою традиційним методам глибокого машинного навчання. Нейроеволюційний алгоритм автоматично розвиває нейронні мережі для конкретного завдання та середовища. Перевага полягає в тому, що необхідно лише абстрактно визначити бажану поведінку, а алгоритм максимально оптимізує штучну нейронну мережу для виконання вимог. Розглянутий алгоритм NEAT дозволяє обробляти багатовимірні простори стану та дії, забезпечуючи гнучкість для керування складними рухами руки робота. Він також здатний виявляти політики керування, які демонструють непередбачувану поведінку, явно не запрограмовану людьми – інженерами. Нейроеволюція дозволяє паралельно оцінювати кілька нейронних мереж, забезпечуючи ефективне дослідження простору пошуку. Роботу алгоритму досліджено у експерименті, проведеному у двовимірному середовищі з роботизованою рукою для задачі позиціонування.

Ключові слова: нейроеволюція, маніпулювання об'єктами, контролер роборуки.

USE OF NEUROEVOLUTION FOR NEURAL NETWORK POLICIES SEARCH FOR ROBOTIC ARM

A. Vitiuk[1], A. Doroshenko[2]
[1,2]National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
37, Prosp. Peremohy, Kyiv, Ukraine, 03056
[2]Institute of Software Systems of the National Academy of Sciences of Ukraine
40, Akademika Glushkova Avenue, Kyiv, Ukraine, 03187
[1]alyonavityuk@gmail.com
[2]doroshenkoanatoliy2@gmail.com
[1]https://orcid.org/0000-0002-1445-9598
[2]https://orcid.org/0000-0002-8435-1451

Abstract. An approach to using neuroevolution to find neural network policies for the task of positioning a robotic arm is considered. As a rule, robotic problems have relatively large solution spaces, so here neuroevolutionary algorithms are a good alternative to traditional methods of deep machine learning. A neuroevolutionary algorithm automatically develops neural networks for a specific task and environment. The advantage is that it is only necessary to define the desired behavior abstractly, and the algorithm optimizes the artificial neural network as much as possible to fulfill the requirements. The considered NEAT algorithm allows processing multidimensional state and action spaces, providing flexibility to control complex robot arm movements. It is also capable of detecting control policies that exhibit unpredictable behavior that is not clearly programmed by human engineers. Neuroevolution allows multiple neural networks to be evaluated in parallel, providing efficient exploration of the search space. The operation of the algorithm was investigated in an experiment conducted in a two-dimensional environment with a robotic arm for the positioning task.

Keywords: neuroevolution, objects manipulation, roboarm controller.

Вступ

В останні роки машинне навчання зробило революцію у сфері робототехніки та автоматизації. Використовуючи алгоритми, роботів можна навчити виконувати різні завдання і навіть навчатися самостійно. Це дозволило створити більш досконалих роботів, які можуть самостійно орієнтуватися в складних середовищах, взаємодіяти з людьми більш природним чином, а також виконувати виробничі завдання більш ефективно.

Машинне навчання дозволяє роботам обробляти величезні обсяги даних у реальному часі, дозволяючи їм приймати більш швидкі та точні рішення. Це допомагає роботам краще розуміти навколишнє середовище та об’єкти навколо них. Наприклад, їх можна запрограмувати ідентифікувати об’єкти за допомогою комбінації візуальних, тактильних і звукових датчиків. Це дозволяє розпізнавати різні об’єкти в навколишньому середовищі та відповідно реагувати.

Машинне навчання також дозволяє роботам вчитися на їхньому досвіді. Використовуючи дані, зібрані з попередніх завдань, роботи можуть коригувати свою поведінку, щоб краще виконувати майбутні завдання. Це особливо корисно в областях, де роботам потрібно швидко адаптуватися до мінливих умов і середовища. Більшість робототехнічних завдань вимагають багатокрокових цілей, які вимагають механізму планування. Традиційні роботизовані контролери зазвичай проєктуються вручну, а розподіл завдань на підзадачі виконує людина [1].

Багато сценаріїв вимагають адаптації роботів до нових умов або навіть навчання абсолютно новій поведінці. Роботу, який виробляє автомобілі, наприклад, час від часу доведеться адаптуватися до нових моделей автомобілів. Для багатьох реальних додатків достатньо запрограмувати необхідну поведінку вручну, але часто це неможливо, оскільки середовище може просто змінюватися надто часто або навіть бути заздалегідь невідомим інженерам, які програмують систему.

Методи навчання та еволюційні техніки призначені для оптимізації швидкої реакції, яка потім використовується деяким контролером вищого рівня. Оскільки завдання та робоче середовище для роботів стають все більш складними, зростає потреба в методах навчання та пошуку, які можуть планувати досягнення мети, не покладаючись на вже існуючу структуру підзадач, розроблених людиною.

Оскільки традиційні методи глибокого навчання досить обмежені, все більше дослідників почали шукати альтернативні підходи до навчання штучних нейронних мереж. Глибоке машинне навчання надзвичайно потужне для розпізнавання образів, але не дозволяє виконувати завдання, які вимагають розуміння контексту або працюють з незнайомими даними. Багато дослідників сходяться на думці, що сучасний підхід до проєктування систем штучного інтелекту вже не в змозі впоратися з актуальними проблемами.

Альтернативою традиційним методам глибокого машинного навчання є нейроеволюційні алгоритми. Нейроеволюція — це сімейство методів машинного навчання, які використовують еволюційні алгоритми для полегшення вирішення складних проблем, таких як ігри, робототехніка та моделювання природних процесів. Нейроеволюційні алгоритми імітують процес природного відбору. Кінцевим результатом нейроеволюції є оптимальна топологія мережі, яка робить модель більш ресурсоефективною та легшою для аналізу [4].

Нейроеволюційний алгоритм автоматично розвиває нейронні мережі для конкретного завдання та середовища. Перевагою є те, що необхідно лише абстрактно визначити бажану поведінку, а алгоритм максимально оптимізує штучну нейронну мережу для виконання вимог. Однак застосувати нейронні мережі для робототехніки все ще складно, і досягнута продуктивність часто нижча, ніж очікується. Оскільки нейронні мережі

загалом і нейроеволюційні алгоритми зокрема мають багато переваг і недоліків порівняно з альтернативами, цікаво розглянути їх порівняння з іншими підходами [9]. Дослідження показують, що все ще досить складно використовувати штучні нейронні мережі в задовільний спосіб для контролю стану автономного робота [5]. Щоб отримати прийнятні результати від генетичних алгоритмів, потрібно багато часу. Оскільки штучні нейронні мережі прямого зв'язку не мають жодного внутрішнього стану, їх обчислювальні можливості дуже обмежені, що може зробити їх гіршими від конкуруючих підходів. Однак нейроеволюція є абсолютно загальним підходом, який можна застосовувати без будь-яких детальних знань про навколишнє середовище. Крім того, цей клас алгоритмів може адаптуватися до непередбачених змін у середовищі [6]. При проєктуванні контролерів спеціального призначення для досягнення прийнятних результатів потрібна детальна інформація про задачу та середовище. Крім того, важко або навіть неможливо розробити ці контролери таким чином, щоб вони могли адаптуватися до будь-яких змін у середовищі.

Таким чином, нейроеволюцію варто використовувати, якщо знання про задачу дуже обмежені або якщо задача може змінитися непередбачуваним чином у майбутньому. Проте, якщо середовище та задача добре визначені, написання алгоритму з нуля є кращим підходом. У таблиці 1 наведено огляд переваг і недоліків двох підходів.

Дана робота присвячена розробці засобів автоматизації проєктування нейроконтролера для управління роботизованою кінцівкою. Точне розташування руки робота дозволяє точно та повторювано виконувати завдання, що призводить до підвищення ефективності та продуктивності. При правильному позиціонуванні роботи можуть виконувати завдання швидше, з меншою кількістю помилок і цілодобово, що значно економить час і кошти. Точне розташування кінцівки має важливе значення для виконання таких завдань, як обробка деталей, складання, зварювання або фарбування. Позиціонування руки робота має важливе значення для роботи в складних і динамічних середовищах. Точно розташовані руки робота можуть адаптуватися до мінливих умов, уникати перешкод і переміщатися складними шляхами. Це особливо важливо в таких галузях, як логістика, складське господарство та охорона здоров'я, де роботам потрібно взаємодіяти з динамічним і незнайомим середовищем.

Таблиця 1. Порівняння нейроеволюції та контролерів спеціального призначення [5]

Підхід	Переваги	Недоліки
Нейроеволюція	Не потрібні детальні знання про проблему. Агенти адаптуються до змін середовища	Непередбачувані та суперечливі результати. Симуляції з інтенсивними обчисленнями.
Контролери спеціального призначення	Зазвичай більш висока продуктивність. Результати передбачувані та зрозумілі.	Необхідне детальне знання проблеми. Зазвичай не в змозі адаптуватися до змін навколишнього середовища.

1. Політики позиціонування роботизованої руки

Позиціонування роботизованої руки – це процес точного керування положенням і орієнтацією кінцівки роботизованої руки (інструмента або захватного пристрою) для виконання певних завдань. Це положення має вирішальне значення для взаємодії руки з об'єктами, маніпулювання ними та точного виконання різних операцій.

Роботизована рука зазвичай складається з кількох шарнірів, які забезпечують руку ступенями свободи (DOF). DOF представляє кількість незалежних параметрів, необхідних для опису конфігурації руки. Кожен суглоб дозволяє руці обертатися або переміщатися вздовж певної осі, дозволяючи руці рухатися в кількох напрямках. Позиціонування роботизованої руки базується на використанні систем координат для визначення положення та орієнтації руки. Найпоширенішою

системою координат є декартова система координат, де положення визначаються координатами X, Y і Z. Орієнтація руки може бути представлена за допомогою кутів Ейлера, кватерніонів або матриць обертання.

Пряма кінематика передбачає визначення положення та орієнтації кінцевого ефектора на основі кутів або довжин з'єднань. З іншого боку, зворотна кінематика передбачає знаходження кутів або довжин з'єднань, необхідних для досягнення бажаного положення та орієнтації кінцівки.

Планування траєкторії передбачає створення плавного й оптимального шляху, яким рука робота буде рухатися під час переходу з одного положення в інше. Він враховує такі фактори, як уникнення перешкод, спільні обмеження та оптимізація шляху на основі таких критеріїв, як час, споживання енергії або точність. Планування траєкторії гарантує, що рука рухається ефективно та безпечно, досягаючи бажаного положення.

Для точного позиціонування роботизованих рук використовуються різні методи керування. При управлінні з відкритим циклом попередньо визначені команди надсилаються на руку без зворотного зв'язку, припускаючи, що рука рухатиметься згідно з інструкціями. Контроль із замкнутим контуром, також відомий як контроль із зворотним зв'язком, включає зворотний зв'язок датчика для постійного регулювання положення руки та коригування будь-яких відхилень.

Для задачі навчання ми розглянемо модель роботизованої руки, представлену як інтелектуальний агент в обмеженому та спрощеному середовищі. Агент — це суб'єкт, який здатний сприймати своє оточення за допомогою датчиків, а також впливати на своє оточення за допомогою приводів. Агенти зазвичай можуть сприймати власні дії, але не обов'язково вплив цих дій на навколишнє середовище. Агент може бути математично описаний функцією агента, яка визначає дію агента на основі всієї його послідовності сприйняття. Таким чином, поведінка агента може бути повністю описана шляхом визначення дії для кожної можливої послідовності сприйняття [10]. Схема взаємодії агента і середовища представлена на рис. 1.

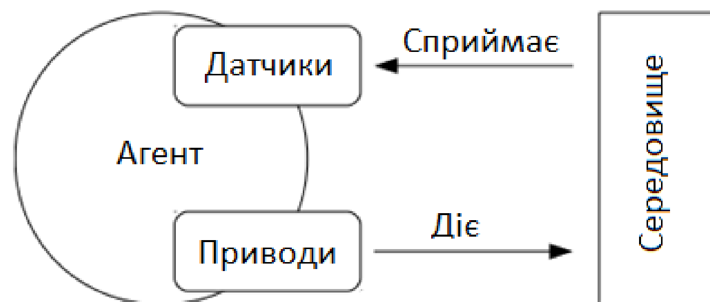


Рис. 1. Взаємодія компонентів навчального середовища

Взаємодія з цільовим об'єктом потребує певного планування. Для успішної маніпуляції об'єктом завдання буде розділено на дві підзадачі з окремими контролерами. Перша підзадача полягає в тому, щоб перемістити кінцівку відносно близько до цільової позиції, уникаючи перешкод. Друга підзадача відноситься до прямого маніпулювання об'єктом, наприклад переміщення випадково розміщених об'єктів із початкового стану до заданої цільової конфігурації. Необхідність використання двох контролерів для цього завдання демонструє, що нейроеволюції легше розвинути реактивну поведінку, ніж довгострокову стратегічну поведінку.

Давайте окремо розглянемо першу визначену підзадачу – позиціонування кінцівки роботизованої руки за допомогою середовища OpenAI Gym із двовимірною роботизованою рукою з двома суглобами.

Це середовище моделювання, призначене для забезпечення віртуальної платформи для тестування та розробки алгоритмів керування для двовимірної роботизованої руки [7]. Приклад стану навколишнього середовища представлений на рис. 2.

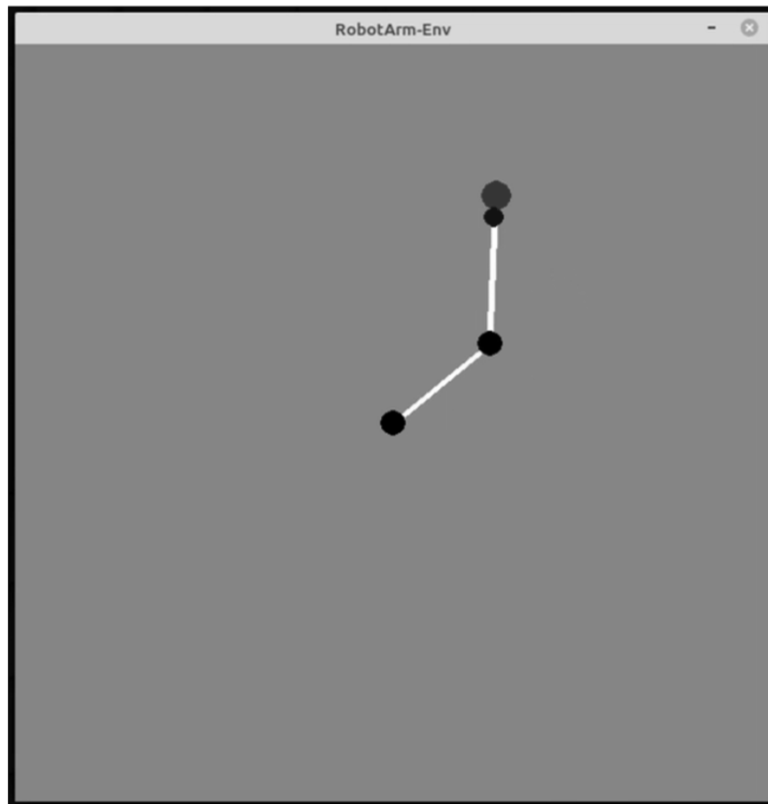


Рис. 2. Середовище OpenAI Gym 2D Robot Arm

Робот складається з двох з'єднань, кожне довжиною 100 пікселів, і мета — досягти червоної крапки, яка випадковим чином генерується в кожному епізоді. Простір дій представляє можливі дії, які може виконати агент. У двовимірному середовищі маніпулятора простір дій складається з керування кутами з'єднання маніпулятора робота:

– 0: залишити поточне значення кута з'єднання;
– 1: приріст кута з'єднання 1;
– 2: зменшення кута з'єднання 1;
– 3: приріст кута з'єднання 2;
– 4: зменшення кута з'єднання 2;
– 5: приріст кутів з'єднань 1 і 2;
– 6: зменшення кутів з'єднань 1 і 2.

Простір спостережень дає інформацію про поточний стан навколишнього середовища. У середовищі двовимірної руки робота простір спостережень є безперервним і містить відповідну інформацію, необхідну для контролю та прийняття рішень:

– цільова позиція в напрямку x (у пікселях);
– цільова позиція в напрямку y (у пікселях);
– поточне положення з'єднання 1 (у радіанах);
– поточне положення з'єднання 2 (у радіанах).

Навколишнє середовище забезпечує винагороду для управління процесом навчання. Винагорода залежить від досягнення цільової позиції:

– робот отримає штраф -1, якщо поточна відстань між кінцівкою і цільовою позицією більша, ніж попередня відстань;
– робот отримає винагороду 1, якщо поточна відстань між кінцівкою і цільовою позицією > -Ɛ < Ɛ, де Ɛ = 10 пікселів.

Крім того, визначено умови завершення, щоб вказати кінець епізоду: поточна винагорода становить -10 або +10.

2. Нейроеволюційне навчання

Навчання з підкріпленням (RL) було успішно застосовано до політик контролю навчання для рук роботів. Алгоритми RL дозволяють роботам вивчати оптимальні стратегії керування методом проб і помилок, взаємодіючи з навколишнім середовищем і отримуючи зворотній зв’язок у формі винагород або штрафів.

У RL ми тренуємо політики агентів π_n, n = 1, ..., N для вирішення окремих випадків середовища: для кожного кроку середовища t агент спостерігає стан середовища s_t, і політика вирішує, яка дія a_t має бути здійснена. Агент отримує винагороду $r(s_t, a_t, s_{t+1})$ за кожен спостережуваний стан під час епізоду навчання. Це призводить до траєкторії з T кроків. Сукупна винагорода $\sum_{k=0}^{T} \gamma^k r_{t+k+1}$ наприкінці епізоду, дисконтована за коефіцієнтом γ, зазвичай використовується як цільова функція політики процесу оптимізації. Називатимемо R_t повного епізоду пристосованістю політики [2].

Алгоритми RL навчаються через ітераційний процес взаємодії з середовищем. Рука робота виконує дії на основі поточної політики, спостерігає за кінцевим станом і винагородою та відповідно оновлює свою політику. Цей процес триває, доки агент не наблизиться до оптимальної політики контролю, яка максимізує сукупну винагороду з часом. Навчання можна проводити онлайн, коли рука навчається під час взаємодії з фізичним середовищем, або офлайн, використовуючи попередньо зібрані дані чи симуляції. Використовуючи навчання з підкріпленням руки роботів можуть автономно навчатися політикам керування, які дозволяють їм виконувати складні завдання, адаптуватися до змінюваних середовищ і навіть вчитися на прикладі людей. Політики керування на основі RL були успішно застосовані до широкого діапазону задач, зокрема маніпулювання об’єктами, складання, захвата та промислової автоматизації.

Нейронна мережа обрана як політика керування роботом. Мережева архітектура зазвичай складається з вхідних нейронів, що представляють стан руки робота, прихованих нейронів для проміжних обчислень і вихідних нейронів, які генерують керуючі сигнали. Вагові коефіцієнти та зміщення нейронної мережі є параметрами, які потрібно оптимізувати. Під час кожного покоління нейронні мережі оцінюються на основі їх продуктивності в попередньо визначеній функції відповідності. Фітнес-функція вимірює, наскільки добре рука робота виконує певне завдання, наприклад досягає цілі, маніпулює об’єктами або уникає перешкод. Оцінка придатності може бути проведена шляхом моделювання або фізичних експериментів, залежно від наявних ресурсів.

Еволюційні алгоритми використовують різні оператори, такі як відбір, кросовер і мутація, щоб створити нові покоління нейронних мереж [1]. Відбір надає перевагу найкращим мережам, що дозволяє їм передавати свої гени наступному поколінню. Схрещування поєднує параметри двох батьківських мереж для створення нащадків із сумішшю їхніх ознак. Мутація вносить невеликі випадкові зміни в параметри, сприяючи дослідженню простору пошуку [3].

Процес нейроеволюції ітеративно розвиває популяцію нейронних мереж протягом кількох поколінь. Оцінка придатності, відбір, схрещування та мутація повторюються, доки не буде знайдено задовільну політику контролю. Процес спрямований на збіжність до параметрів нейронної мережі, які надають політики керування з вищими значеннями пристосованості, ефективно покращуючи продуктивність руки робота [8].

NEAT (NeuroEvolution of Augmenting Topologies) — це нейроеволюційний алгоритм, спеціально розроблений для розвитку штучних нейронних мереж (ANN) зі складною та змінною топологією. Він був представлений Кеннетом О. Стенлі та Рісто Мійкулайненом у 2002 році і з тих пір став популярним підходом у галузі нейроеволюції [3].

NEAT вирішує проблеми розвитку ШНМ із різною кількістю нейронів і з’єднань. На відміну від традиційних методів еволюції нейронних мереж, які спираються на фіксовані топології, NEAT дозволяє створювати та розвивати нові мережеві структури.

Підхід використовує історичну схему маркування, щоб забезпечити збереження інновацій у структурі мережі під час процесу еволюції. Кожному гену в геномі присвоюється історичний маркер для відстеження його походження та збереження історичного запису інновацій. Це дозволяє зберігати та рекомбінувати нові структурні особливості протягом поколінь. Він включає видоутворення для заохочення збереження різноманітності в популяції. Під час еволюції особини групуються у види на основі їх подібності в структурі мережі. Видоутворення допомагає запобігти втраті перспективних структур через надмірну конкуренцію та дозволяє досліджувати різні топології.

Алгоритм дозволяє мережам рости та збільшувати складність протягом поколінь. Він поступово вводить нові вузли та з’єднання, починаючи з простих структур і поступово ускладнюючи їх у процесі еволюції. Крім того, NEAT підтримує спрощення мереж при виявленні зайвих або непотрібних з'єднань, покращуючи ефективність і продуктивність.

NEAT був успішно застосований у різних областях, включаючи системи керування, ігри, робототехніку та розпізнавання образів. Його здатність працювати з динамічними та змінними мережевими топологіями робить його добре придатним для завдань, які потребують адаптивності, масштабованості та відкриття нових мережевих архітектур.

Підхід поєднання еволюційних алгоритмів із нейронними мережами надихнув на подальші розробки в галузі нейроеволюції. Розширення та варіації NEAT були запропоновані для вирішення конкретних завдань, таких як мережі створення композиційних шаблонів (CPPN) для генеративних завдань або HyperNEAT для розвитку великомасштабних нейронних мереж. Загалом він виявився потужним і гнучким нейроеволюційним алгоритмом, що забезпечує еволюцію складних структур нейронної мережі та просуває сферу еволюційної робототехніки та штучного інтелекту.

3. Застосування підходу для задачі позиціонування робочої кінцівки робота

Для експерименту була обрана бібліотека NEAT-Python як реалізація алгоритму NEAT. Як випливає з назви, це реалізація алгоритму NEAT на мові програмування Python. Бібліотека NEAT-Python забезпечує реалізацію стандартних методів NEAT для моделювання генетичної еволюції геномів організмів у популяції. Вона містить утиліти для перетворення генотипу організму в його фенотип (штучну нейронну мережу) і надає зручні методи для завантаження та збереження конфігурацій геному разом із параметрами NEAT. Крім того, вона надає корисні процедури, які допомагають збирати статистичні дані про хід еволюційного процесу та зберігати/завантажувати проміжні контрольні точки. Контрольні точки дозволяють нам періодично зберігати стан еволюційного процесу і пізніше відновлювати виконання процесу [10].

У процесі нейроеволюції використовується двовимірний симулятор роботизованої руки для реалізації методу навчання методом спроб і помилок. Він підтримує популяцію геномів, які розвиваються з покоління в покоління, доки не буде знайдено успішний роз’вязок. У процесі еволюції кожен організм у популяції перевіряється на придатність шляхом імітації руки з двома суглобами. Наприкінці симуляції організм отримує сигнал винагороди у вигляді кількості часових кроків, протягом яких він міг досягти цільової червоної точки та максимальної винагороди. Отриманий сигнал винагороди визначає працездатність організму і вирішує його долю в процесі нейроеволюції.

Експеримент підкреслив важливість підтримки збалансованої популяції із помірною кількістю видів. Занадто велика різноманітність видів у популяції може перешкоджати процесу нейроеволюції,

зменшуючи ймовірність спарювання між організмами, що належать до різних видів. Крім того, враховуючи, що чисельність популяції фіксована, чим більше видів у популяції, тим менша чисельність кожного виду. Менші види зменшують ймовірність корисних мутацій. З іншого боку, наявність окремих видів дозволяє нам підтримувати корисні мутації в кожній ніші видоутворення та використовувати кожну мутацію в наступних поколіннях. Таким чином, занадто низька видова різноманітність також шкодить еволюції.

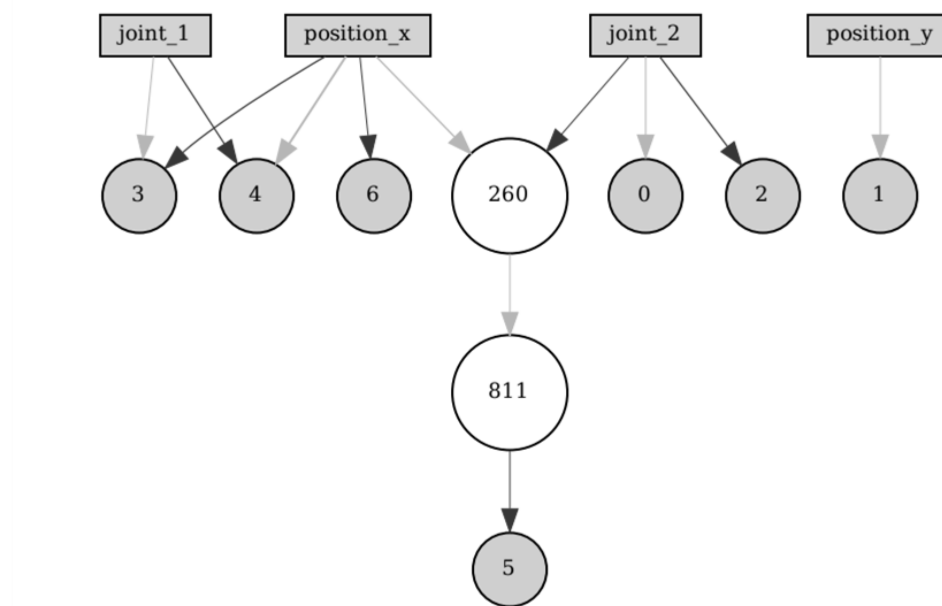


Рис. 3. Нейронна мережа після 1000 поколінь

Використання нейроеволюції в пошуку політики для позиціонування роботизованої руки має враховувати компроміс між трьома параметрами системи, що розробляється:

– обсяг тренувальних зусиль, необхідних для виконання завдання за допомогою робота;

– час на виконання завдання після отримання команди управління;

– якість результату виконаного завдання (точність позиціонування).

Висновки

Розглянутий підхід показує перспективність застосування нейроеволюції для автоматизації розробки нейронних мереж для задач управління роботизованими системами з великими просторами рішень.

Нейроеволюція в управлінні рукою робота має кілька переваг. Вона може обробляти багатовимірні простори стану та дії, забезпечуючи гнучкість для керування складними роботами. Вона також здатна виявляти політики керування, які демонструють непередбачувану поведінку, явно не запрограмовану інженерами. Нейроеволюція дозволяє паралельно оцінювати кілька нейронних мереж, забезпечуючи ефективне дослідження простору пошуку.

Однак нейроеволюція також стикається з проблемами. Вона може бути обчислювально затратною, особливо при роботі з великими нейронними мережами та складними завданнями.

Використовуючи переваги NEAT, можливо використовувати еволюційні процеси для розвитку нейронних мереж, здатних до точного та адаптивного керування позиціонуванням роботизованої руки. Гнучкі топології, можливості оптимізації та здатність обробляти складну динаміку сприяють підвищенню продуктивності, адаптивності та ефективності систем позиціонування роботизованої руки.

Література

1. R. Mahjourian, R. Miikkulainen, Neuroevolutionary Planning for Robotic Control, Department of Computer Science The University of Texas at Austin Austin, 2018.

2. Stork, Jörg, Zaefferer et al., Behavior-based Neuroevolutionary Training in Reinforcement Learning, 2021. doi:10.48550/arXiv.2105.07960.

3. Kenneth O. Stanley and Risto Miikkulainen, Evolving Neural Networks Through Augmenting Topologies, Evolutionary Computation 10 (2): 99-127, 2002.

4. Застосування засобів нейроеволюції в технічних системах автоматизації керування / А.Ю. Дорошенко, І.З. Ашур // Проблеми програмування. — 2021. — № 1. — С. 16-25.

5. M. Wurtinger, Neuroevolution for Robot Control, Test Framework and Experimental Evaluation, Institut fur Informatik Lehrstuhl fur Programmierung und Softwaretechnik, 2011. URL:https://www.pst.ifi.lmu.de/Lehre/Abschlussarbeiten/vorlagen/thesis-wuertinger_2011-12-19.pdf.

6. OpenAI, et al. Asymmetric self-play for automatic goal discovery in robotic manipulation. arXiv preprint arXiv:2101.04882, 2021.

7. OpenAI Gym 2D Robot Arm Environment, URL: https://github.com/ekorudiawan/gym-robot-arm.

8. Huang, P.-C., Lehman et al., Grasping novel objects with a dexterous robotic hand through neuroevolution. 2014 IEEE Symposium on Computational Intelligence in Control and Automation (CICA). doi:10.1109/cica.2014.7013242.

9. Huang, Pei-Chi, Sentis et al., Tradeoffs in Neuroevolutionary Learning-Based Real-Time Robotic Task Design in the Imprecise Computation Framework. ACM Transactions on Cyber-Physical Systems. 3. 1-29. 10.1145/3178903, 2019.

10. Omelianenko, Hands-On Neuroevolution with Python: Build high-performing artificial neural network architectures using neuroevolution-based algorithms, Packt Publishing, 2019.

References

1. R. Mahjourian, R. Miikkulainen, Neuroevolutionary Planning for Robotic Control, Department of Computer Science The University of Texas at Austin Austin, 2018.

2. Stork, Jörg, Zaefferer et al., Behavior-based Neuroevolutionary Training in Reinforcement Learning, 2021. doi:10.48550/arXiv.2105.07960.

3. Kenneth O. Stanley and Risto Miikkulainen, Evolving Neural Networks Through Augmenting Topologies, Evolutionary Computation 10 (2): 99-127, 2002.

4. A.Yu. Doroshenko, I.Z. Achour, Application of neuro evolution tools in automation of technical control systems, Prombles in programming 2021; 1: 16-25. doi:10.15407/pp2021.01.016.

5. M. Wurtinger, Neuroevolution for Robot Control, Test Framework and Experimental Evaluation, Institut fur Informatik Lehrstuhl fur Programmierung und Softwaretechnik, 2011. URL:https://www.pst.ifi.lmu.de/Lehre/Abschlussarbeiten/vorlagen/thesis-wuertinger_2011-12-19.pdf.

6. OpenAI, et al. Asymmetric self-play for automatic goal discovery in robotic manipulation. arXiv preprint arXiv:2101.04882, 2021.

7. OpenAI Gym 2D Robot Arm Environment, URL: https://github.com/ekorudiawan/gym-robot-arm.

8. Huang, P.-C., Lehman et al., Grasping novel objects with a dexterous robotic hand through neuroevolution. 2014 IEEE Symposium on Computational Intelligence in Control and Automation (CICA). doi:10.1109/cica.2014.7013242.

9. Huang, Pei-Chi, Sentis et al., Tradeoffs in Neuroevolutionary Learning-Based Real-Time Robotic Task Design in the Imprecise Computation Framework. ACM Transactions on Cyber-Physical Systems. 3. 1-29. 10.1145/3178903, 2019.

10. Omelianenko, Hands-On Neuroevolution with Python: Build high-performing artificial neural network architectures using neuroevolution-based algorithms, Packt Publishing, 2019.

The article has been sent to the editors 17.08.23.
After processing 25.08.23.
Submitted for printing 30.08.23.